THINK INDIA JOURNAL

ISSN: 0971-1260
Vol-22-Issue-16-August-2019

Software Defects Prediction Using Support Vector Machine

Kiran Kumar Kaki

School Of Computer Science and Engineering

Lovely Professional University

Email:kiran.kaki@lpu.co.in

Abstract:

In the field of software engineering, there are a number of prediction methods that are fault estimation, security estimation, effort estimation, correlation cost estimation, re-usability estimation, test effort estimation and quality estimation. These methods are helps to reduce the cost of testing which ultimately reduce the cost of the project. One of the most important stage is testing of software which reduces the defects of the software. Software defects are predicted by a software developers or testers at an early stage of software development life cycle and it reduces the overall time, cost and effort of the software development team. Nowadays, defect prediction methods are based on an adequate quantity of historic project data. Many researchers are using NASA's PROMISE dataset to develop prediction techniques for various phases of software development. We use the dataset available for defect prediction. This work aims to achieve high prediction accuracy by applying Support Vector Machine based technique. In this research work, our primary target will be to focus on making a MATLAB based interface to predict software defects. This interface will implement SVM as the underlying algorithm and will be trained and tested using ANT-1.7 dataset.

1. Introduction

Software metrics are utilized to discover the Software faults in the Software improvement life cycle before the testing procedure. Routines utilized are insights, machine learning, machine adapting alongside measurable techniques and factual models versus expert estimation [1]. Defects in programming frameworks prompts significant issues in software. A large portion of programming frameworks are conveyed to clients with intemperate issues. To discover the defect-prone parts of the software and to focus on those parts for expanded quality control and testing is a compelling way to deal with decrease of defects. There are different characteristics like deformity thickness, viability, issue inclination, standardized revamp and re-convenience which focus the nature of the product.

ISSN: 0971-1260
Vol-22-Issue-16-August-2019

Support Vector Machine (SVM) is theoretically a very good algorithm. Vapnik & Chervonenkis (1960s) developed SVM from SLT. In SVM, data is being differentiated into 2 sets, training and testing. Every record in the training set contains 1 target value or class name and contains a few properties known as watched variables. SVM discovers a direct dividing hyper-plane. The equation for partition is C=AX+BY. SVM is utilized as a part of numerous fields. SVM is utilized as a part of twofold arrangement errands. SVMs are another promising non-direct, non-parametric order method. SVM is utilized as a part of the restorative diagnostics, optical character recognition, electric load anticipating and other numerous fields.

In this paper, we propose a multi-class SVM based technique to predict software defects. The proposed technique has been implemented into a simple yet effective graphical user interface to let the user easily input the required parameters and predict the number of software defects that may arise in the future. This detection leads to a very convenient phase in the software development life cycle, where the user can make necessary changes in the software and/ or the methodology being applied so that the faults can be avoided at later stages. Hence, it can save a huge sum in the overall development cost of the software. For training and testing purposes, dataset from NASA namely ANT 1.7 has been used. The detail of the dataset has been given in the forthcoming sections of this paper. Experiments have shown a very encouraging 89.0909% of accuracy in the prediction results.

2. Literature Review

Scientists can also apply a machine learning method to RL-Software system to predict the software faults. Types include telecontrol / telepresence, automation, and applications for mission planning.

Only 31 software development projects were submitted by Fenton et al.[2]. This informationset integrates the collection of quantitative and qualitative variables previously integrated into the computer process's causal model. The variables include values for software width, commitment and errors, as well as subjective information values measured using a questionnaire by project managers. They used these data to test the concept of causation.

Design[3] and software metrics[4] are used to assess the reliability of predictive defect models available before and after implementation of the program. Code metrics and development

ISSN: 0971-1260
Vol-22-Issue-16-August-2019

metrics are only present after implementation of the program and before beginning the coding[5]. Models are based on data from a single release of Ericsson's built broad telecommunications framework using linear regression.

Specialists use mathematical techniques and methods of machine learning to predict the code's weakness in their applications. The execution of Lines of Code metric is strong in their analysis and the accuracy of Lack of Cohesion on Methods metric is great but its culmination value is poor[6].

Song et al. propose and evaluate a general computer defect prediction model that facilitates an objective and detailed analysis of competing prediction systems[7].

Koru and Liu show that static measurements and class-level defect information can be used to construct machine-learning models that predict top-class defects in practice[8].

3. Proposed Methodology

We propose the use of Multi-class Support Vector Machine (McSVM) to classify the input set of parameters into one of the several classes to estimate the number of defects that may arise in software that is under construction. Figure 1 shows classification in case of multi class SVM.

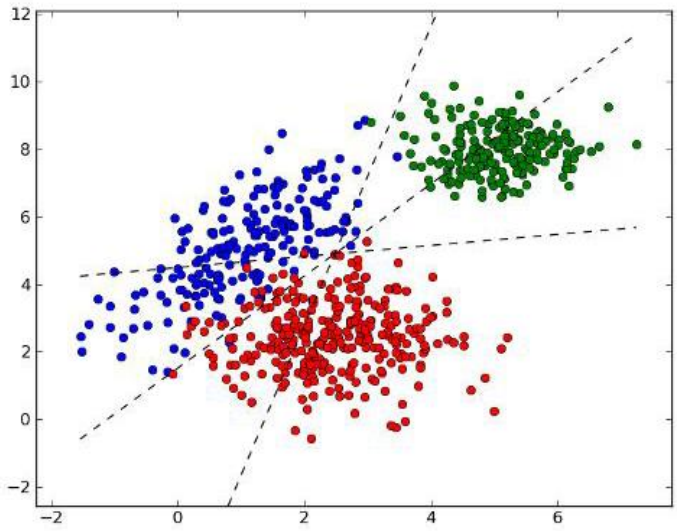

Fig. 1: Classification in Multi Class SVM.

ISSN: 0971-1260
Vol-22-Issue-16-August-2019

The implementation of McSVM uses a multi-classification strategy "all-together," which is computationally more costly but typically more effective than multi-classification approaches "one-against-all" or "one-against-one." Comparisons of these techniques were not made to large-scale problems. Particularly for methods that find more than one-class SVM in single stage, a much greater problem of optimization is needed, so experiments are restricted to tiny data samples up to now.

In the present research work, we use ANT 1.7 dataset given by NASA's Promise dataset repository. The dataset contains 745 records. We divided the dataset into two subsets; training set containing 635 records (approx. 85% of total) and testing dataset containing 110 records (approx. 15% of total). These two datasets are used to train and test the tool, respectively. Each record of the dataset provides us with 18 input metrics and 1 target or output parameter

4. Implementation

4.1 User Friendly Tool

A user friendly tool using MATLAB R2015a has been developed. This tool allows the user to train the McSVM. After the training is over, it lets the user input various parameter values to estimate the number of defects that may arise in the software being implemented. To test the accuracy of the proposed system, an option has been given which reads the test dataset and compares the predicted and actual outputs. Figure 2 shows the GUI with various input boxes and buttons to train, predict and test.

ISSN: 0971-1260
Vol-22-Issue-16-August-2019

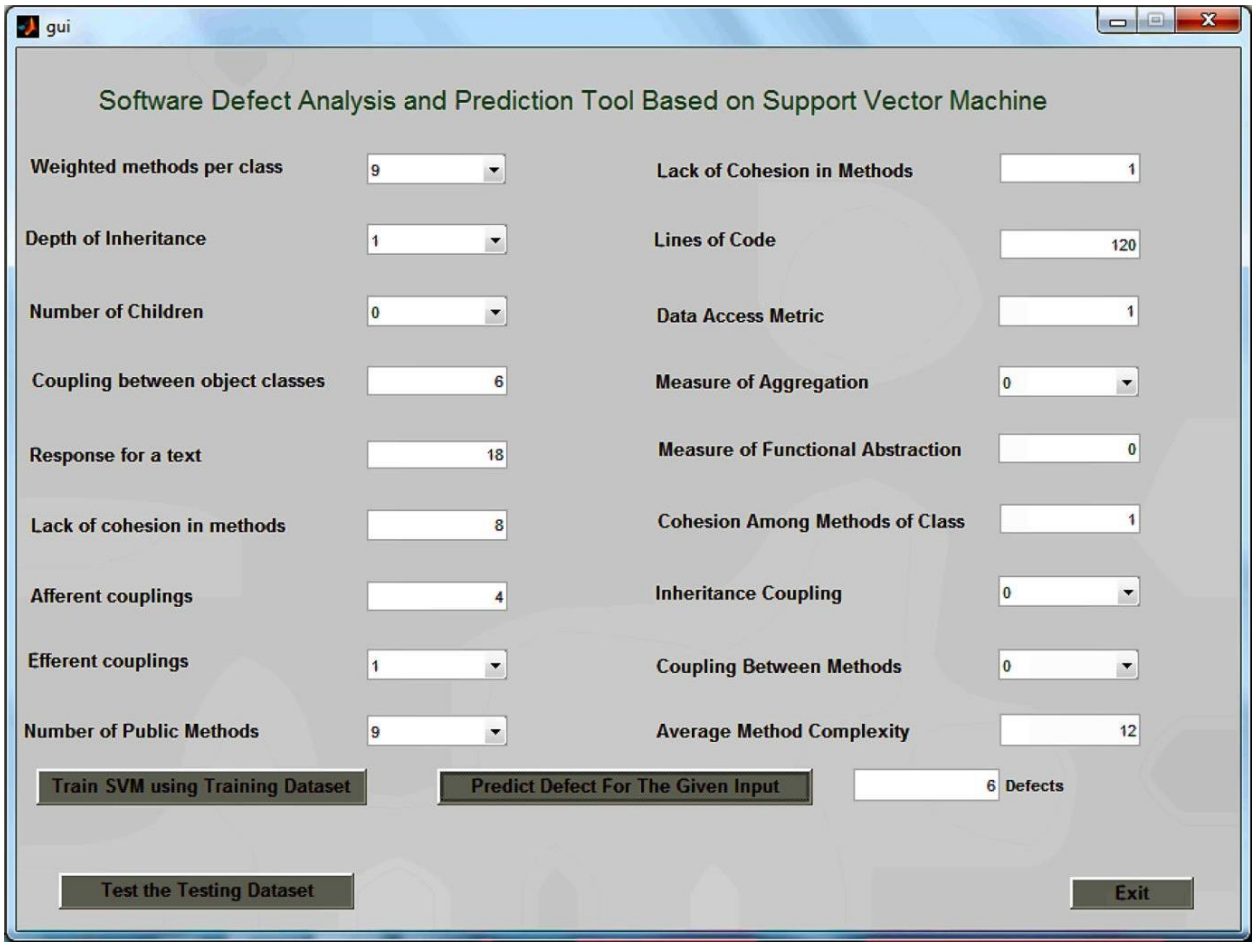

Fig. 2: User Friendly GUI.

4.2 Preparing the data

Data required to train and test the system has been taken from the PROMISE repository. We chose ANT-1.7 dataset. This dataset contains a total of 745 records, out of which 635 (nearly 85%) were used to train the McSVM and rest (110, nearly 15%) were used for testing the system. The selection of record for training and testing was done on a pure random basis.

4.3 Train and then predict the Multiclass SVM

It is necessary to train the McSVM prior to making any use of it. Thus, an option is provided on the panel to train McSVM. There is a file namely, traindata.txt available for the tool to read and get trained. It reads all the 635 training records one by one and accordingly gets trained. Once the McSVM training is finished, the tool is ready for prediction or testing. It is wise to test the tool before deploying it for prediction. Thus, we use the option to test the testing dataset. Again,

ISSN: 0971-1260
Vol-22-Issue-16-August-2019

there is a file namely, testdata.txt available. The tool reads all the records one by one, applies the McSVM classification process and gets the results. These results are then compared with the actual outputs given in each record of the test dataset. Finally, when the authenticity of the tool has been established, it can be put to actual use of prediction. The user may enter the values using textboxes and drop down boxes and can get the predicted result on the panel itself.

5. Experimental Results

Experiments have shown that the prediction accuracy of the proposed technique is quite encouraging. We are able to achieve an accuracy of 89.0909%. As compared to the proposed technique, the accuracy of one of the previously available work is given in table 1. The highest accuracy that it achieved is 71.9%, which is far below the accuracy achieved by the proposed technique.

Technique used	Dataset	Accuracy percentage
Genetic Algorithm with SVM	jE1	71.9%
	jE2	63.7%
	jE1&jE2 (inter-release)	58.0%

Table 1: Accuracy of technique given in.

Table 2 shows the results of testing the test dataset. The last two columns of the table show the predicted number of defects and actual number of defects, respectively.

ISSN: 0971-1260
Vol-22-Issue-16-August-2019

Sr. No.	Defects		Sr. No.	Defects	
	Predicted	Actual		Predicted	Actual
1	0	0	56	0	0
2	0	0	57	0	0
3	0	0	**58**	**4**	**3**
4	0	0	59	0	0
5	0	0	60	0	0
6	0	0	61	0	0
7	**1**	**0**	62	0	0
8	0	0	63	0	0
9	0	0	**64**	**2**	**0**
10	0	0	65	0	0
11	0	0	66	0	0
12	0	0	67	0	0
13	0	0	68	5	5
14	0	0	69	0	0
15	**1**	**0**	70	0	0
16	0	0	71	0	0
17	0	0	72	0	0
18	0	0	73	0	0
19	1	1	74	0	0
20	0	0	75	1	1
21	0	0	76	0	0
22	0	0	77	4	4
23	0	0	78	0	0
24	0	0	**79**	**7**	**5**
25	0	0	80	0	0
26	0	0	81	0	0
27	0	0	82	0	0
28	0	0	83	2	2

Table 2: Comparison of Predicted and Actual Defects in Proposed Technique.

It is observed that in most of the cases, the technique has been accurately able to predict the number of faults. In some of the instance, the results are not as expected. The differences in predicted and actual number of defects are shown in bold. There are a total of 12 differences. Thus, we calculate the error and accuracy percentage as follows:

$$Percentage\ Error = \frac{|Actual\ Outputs - Predicted\ Outputs|}{Actual\ Outputs} X100$$

$$Percentage\ Accuracy = 100 - Percentage\ Error$$

ISSN: 0971-1260
Vol-22-Issue-16-August-2019

Using the above two formulae, we calculate the percentage error as 10.9091% and percentage accuracy as 89.0909%.

6. Conclusion

There may be a variety of software defects. It is important to predict software faults at an early state of software development in order to improve software quality. The primary objective of this topic is to use the multiclass support vector machine to study and understand the principle of prediction of code defects. We used computer metrics in this analysis. This paper shows that 89.0909 percent is the accuracy of the proposed system, which is quite encouraging. Other training algorithms clubbed with the proposed technique may be used in the future to further increase the level of accuracy to predict software defects.

References

[1] Catal, C., & Diri, B. (2009). A systematic review of software fault prediction studies. Expert Systems with Applications, 36(4), 7346-7354.

[2] Fenton, N., Neil, M., Marsh, W., Hearty, P., Radlinski, L., & Krause, P. (2007, May). Project data incorporating qualitative factors for improved software defect prediction. In Proceedings of the Third International Workshop on Predictor Models in Software Engineering (p. 2). IEEE Computer Society.

[3] El Emam, K., Melo, W., & Machado, J. C. (2001). The prediction of faulty classes using object-oriented design metrics. Journal of Systems and Software, 56(1), 63-75.

[4] Zhao, M., Wohlin, C., Ohlsson, N., & Xie, M. (1998). A comparison between software design and code metrics for the prediction of software fault content. Information and Software Technology, 40(14), 801-809.

[5] Tomaszewski, P., Lundberg, L., & Grahn, H. (2005). The accuracy of early fault prediction in modified code. In Proceedings of the Fifth Conference on Software Engineering Research and Practice in Sweden (SERPS) (pp. 57-63).

[6] Gyimothy, T., Ferenc, R., & Siket, I. (2005). Empirical validation of object-oriented metrics on open source software for fault prediction. Software Engineering, IEEE Transactions on, 31(10), 897-910.

ISSN: 0971-1260
Vol-22-Issue-16-August-2019

[7] Song, Q., Jia, Z., Shepperd, M., Ying, S., & Liu, J. (2011). A general software defect-proneness prediction framework. Software Engineering, IEEE Transactions on, 37(3), 356-370.

[8] Koru, A. G., & Liu, H. (2005). Building effective defect-prediction models in practice. Software, IEEE, 22(6), 23-29.